Computing Poetry Style

Rodolfo Delmonte
Department of Language Studies & Department of Computer Science
Ca' Foscari University - 30123, Venezia, Italy
delmont@unive.it

Abstract: We present SPARSAR, a system for the automatic analysis of poetry(and text) style which makes use of NLP tools like tokenizers, sentence splitters, NER (Name Entity Recognition) tools, and taggers. Our system in addition to the tools listed above which aim at obtaining the same results of quantitative linguistics, adds a number of additional tools for syntactic and semantic structural analysis and prosodic modeling. We use a constituency parser to measure the structure of modifiers in NPs; and a dependency mapping of the previous parse to analyse the verbal complex and determine Polarity and Factuality. Another important component of the system is a phonological parser to account for OOVWs, in the process of grapheme to phoneme conversion of the poem. We also measure the prosody of the poem by associating mean durational values in msecs to each syllable from a database and created an algorithm to account for the evaluation of durational values for any possible syllable structure. Eventually we produce six general indices that allow single poems as well as single poets to be compared. These indices include a Semantic Density Index which computes in a wholly new manner the complexity of a text/poem.

Keywords: NLP, Sentiment and Affective Analysis, Factuality and Subjectivity Analysis, Prosodic Structure, Semantic and Syntactic Processing, Metrical Structure

1 Introduction

We present SPARSAR, a system for poetry (and text) style analysis by means of parameters derived from deep poem (and text) analysis. We use our system for deep text understanding called VENSES[6] for that aim. SPARSAR[4] works on top of the output provided by VENSES and is organized in three main modules which can be used also to analyse similarities between couples of poems by the same or different poet and similarities between collections of poems by a couple of poets. These modules produce six general indices which are derived from quantitative evaluation of features derived from the analysis. They include a Semantic Density Index, a Deep Conceptual Index, a Metrical Distance Index, a Prosodic Distribution Index and a Rhyming Scheme Comparison Index. A General Evaluation Index is then produced and used to compare poems and poets with one another and establish a graded list on the basis of the parameters indicated above.
In addition to what is usually needed to compute text level semantic and pragmatic features, poetry introduces a number of additional layers of meaning by means of metrical and rhyming devices. For these reasons more computation is required in order to assess and evaluate the level of complexity that a poem objectively contains. An ambitious project would include computing metaphors and relate the imagery of the poem to his life and his *Weltanschaung*. This is however not our current aim. In particular, as far as metaphors are concerned, we dealt with this topic in another paper [5]. We also dealt with general quantitative measurements of poetic style in the past [2,3].

1.1 State of the Art

Our interest in writing a program for the automatic analysis of poetry style and content derives from D.Kaplan's program called ***American Poetry Style Analyzer***, (hence APSA) for the evaluation and visualization of poetry style. Kaplan's program works on the basis of an extended number of features, starting from word length, type and number of grammatical categories: verb, adjective, noun, proper noun; up to rhythmic issues related to assonance, consonance and rhyme, slant rhyme vs perfect rhyme. The output of the program is a graphic visualization for a set of poems of their position in a window space, indicated by a coloured rectangle where their title is included. D.M.Kaplan worked on a thesis documented in a number of papers [8,9].
I will base my analysis on the collected works of an Australian poet, Francis Webb who died in 1974. Webb was considered one of the best poet in the world at the time of the publication of the first edition of his Collected [11]. In the past, stylistic quantitative analysis of literary texts was performed using concordancers and other similar tools that aimed at measuring statistical distribution of relevant items. Usually adjectives, but also verbs, nouns and proper nouns were collected by manual classification [3,10]. This approach has lately been substituted by a computational study which makes heavy use of NLP tools, starting from tokenizers, sentence splitters, NER (Name Entity Recognition) tools, and finally taggers and chunkers. One such tools is represented by David Kaplan's "American Poetry Style Analyzer" (hence APSA) which was our inspiration and which we intend to improve in our work. We used APSA to compare collected poems of different poets and show the output in a window, where each poet is represented by a coloured rectangle projected in space [5]. The spatial position is determined by some 85 parameters automatically computed by the system on the raw poetic texts. However, the analysis is too simple and naive to be useful and trustful and in fact, a paper by Kao & Jurafsky [7] who also used the tool denounces that. In that paper, Jurafsky works on the introduction of a semantic classifier to distinguish concrete from abstract nouns, in addition to the analysis that the tool itself produces. Kaplan himself denounced shortcomings of his tool when he declared he did not consider words out of the CMU phonetic vocabulary apart from plurals and other simple morphological modifications[1]. Eventually we decided to contribute a much deeper analyzer than the one available by introducing three important and missing factors: phonological rules for OOVWs, and syntax and semantics[2].

2 SPARSAR - Automatic Analysis of Poetic Structure and Rhythm with Syntax, Semantics and Phonology

[1] The syllabified version of the CMU dictionary dates 1998 and is called cmudict.0.6, which is the fifth release of cmudict.
[2] In APSA the position that a poem will take in the window space is computed by comparing values associated automatically to features. The most interesting component of the program is constituted by the presence of weights that can be associated to parameter, thus allowing the system resilience and more perspicuity. Apart from that, there is no way for the user to know why a specific poem has been associated to a certain position in space. This was basically the reason why we wanted to produce a program that on the contrary allowed the user to know precisely why two or more poems were considered alike - on the basis of what attribute or feature - and in which proportion.

SPARSAR[4] produces a deep analysis of each poem at different levels: it works at sentence level at first, than at verse level and finally at stanza level (see Figure 1 below). The structure of the system is organized as follows: at first syntactic, semantic and grammatical functions are evaluated. Then the poem is translated into a phonetic form preserving its visual structure and its subdivision into verses and stanzas. Phonetically translated words are associated to mean duration values taking into account position in the word and stress. At the end of the analysis of the poem, the system can measure the following parameters: mean verse length in terms of msec. and in number of feet. The latter is derived by a verse representation of metrical structure. Another important component of the analysis of rhythm is constituted by the algorithm that measures and evaluates rhyme schemes at stanza level and then the overall rhyming structure at poem level. As regards syntax, we now have at our disposal, chunks and dependency structures if needed. To complete our work, we introduce semantics both in the version of a classifier and by isolating verbal complex in order to verify propositional properties, like presence of negation, computing factuality from a crosscheck with modality, aspectuality – that we derive from our lexica – and tense. On the other hand, the classifier has two different tasks: distinguishing concrete from abstract nouns, identifying highly ambiguous from singleton concepts (from number of possible meanings from WordNet and other similar repositories). Eventually, we carry out a sentiment analysis of every poem, thus contributing a three-way classification: neutral, negative, positive that can be used as a powerful tool for evaluation purposes.
As said above, we have been inspired by by Kaplan's tool APSA, and started developing a system with similar tasks, but which was more transparent and more deeply linguistically-based. The main new target in our opinion, had to be an index strongly semantically based, i.e. a "Semantic Density Index" (SDI). With this definition I now refer to the idea of classifying poems according to their intrinsic semantic density in order to set apart those poems which are easy to understand from those that require a rereading and still remain somewhat obscure. An intuitive notion of SDI can be formulated as follow:
- easy to understand are those semantic structures which contain a proposition, made of a main predicate and its arguments
- difficult to understand are on the contrary semantic structures which are filled with nominal expressions, used to reinforce a concept and are justaposed in a sequence
- also difficult to understand are sequences of adjectives and nominals used as modifiers, union of such items with a dash.
There are other elements that I regard very important in the definition of semantic parameters and are constituted by presence of negation and modality: this is why we compute Polarity and Factuality. Additional features are obtained by measuring the level of affectivity by means of sentiment analysis, focussing on presence of negative items which contribute to make understanding more difficult.
The Semantic Density Index is derived from the computation of a number of features, some of which have negative import while others positive import. At the end of the computation the index may end up to be positive if the poem is semantically "light", that is easy to read and understand; otherwise, it is computed as "heavy" which implies that it is semantically difficult.
At the end we come up with a number of evaluation indices that include: a Constituent Density Index, a Sentiment Analysis Marker, a Subjectivity and Factuality Marker. We also compute a Deep Conceptual Index, see below.

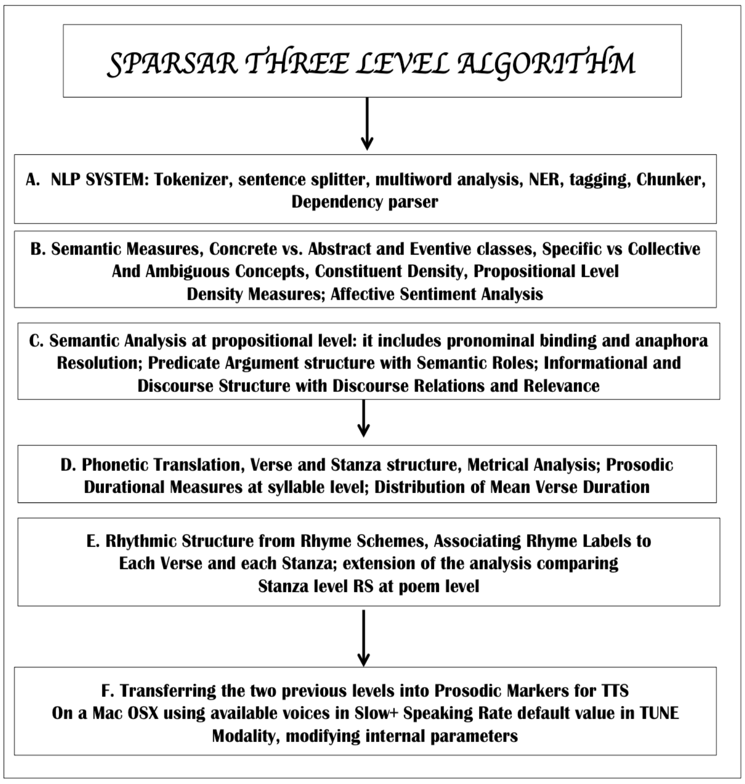
SPARSAR THREE LEVEL ALGORITHM
A. NLP SYSTEM: Tokenizer, sentence splitter, multiword analysis, NER, tagging, Chunker, Dependency parser
B. Semantic Measures, Concrete vs. Abstract and Eventive classes, Specific vs Collective And Ambiguous Concepts, Constituent Density, Propositional Level Density Measures; Affective Sentiment Analysis
C. Semantic Analysis at propositional level: it includes pronominal binding and anaphora Resolution; Predicate Argument structure with Semantic Roles; Informational and Discourse Structure with Discourse Relations and Relevance
D. Phonetic Translation, Verse and Stanza structure, Metrical Analysis; Prosodic Durational Measures at syllable level; Distribution of Mean Verse Duration
E. Rhythmic Structure from Rhyme Schemes, Associating Rhyme Labels to Each Verse and each Stanza; extension of the analysis comparing Stanza level RS at poem level
F. Transferring the two previous levels into Prosodic Markers for TTS On a Mac OSX using available voices in Slow+ Speaking Rate default value in TUNE Modality, modifying internal parameters

Figure 1. The SPARSAR three-level system

The procedure is based on the tokenized sentence, which is automatically extracted and may contain many verses up to a punctuation mark, usually period. Then I use the functional structures which are made of a head and a constituent which are measured for length in number of tokens. A first value of SDI comes from the proportion of verbal compounds and non-verbal ones. I assume that a "normal" distribution for a sentence corresponds to a semantic proposition that contains one verbal complex with a maximum of four non verbal structures. More verbal compounds contribute to reducing the SDI.
The other contribution comes from lemmatization and the association of a list of semantic categories, general semantic classes coming from WordNet or other similar computational lexica. These classes are also called supersense classes. As a criterion for grading difficulty, I consider more difficult to understand a word which is specialized for a specific semantic domain and has only one such supersense label. On the contrary, words or concepts easy to understand are those that are ambiguous between many senses and have more semantic labels associated to the lemma. A feature derived from quantitative linguistic studies is the rare words, which are those words that appear with less than 4 occurrences in frequency lists. I use the one derived from Google GigaWord.
The index will have a higher value for those cases of high density and a lower value for the contrary. It is a linear computation and includes the following features: the ratio of number of words vs number of verbs; the ratio of number of verbal compounds vs non-verbal ones; the internal composition of non-verbal chunks: every additional content word increases

their weight (functional words are not counted); the number of semantic classes. Eventually a single index is associated to the poem which should be able to differentiate those poems which are easy from the cumbersome ones.
What I do is dividing each item by the total number of tagged words and of chunks. In detail, I divide verbs found by the total number of tokens (the more the best); I divide adjectives found by the total number of tokens (the more the worst); I divide verb structures by the total number of chunks (the more the best); I divide inflected vs uninflected verbal compounds (the more the best); I divide nominal chunks rich in components : those that have more than 3 members (the more the worst); I divide semantically rich (with less semantic categories) words by the total number of lemmas (the more the worst); I count rare words (the more the worst); I count generic or collective referred concepts (the more the best); I divide specific vs ambiguous semantic concepts (those classified with more than two senses) (the more the worst); I count doubt and modal verbs, and propositional level negation (the more the worst); I divide abstract and eventive words vs concrete concepts (the more the worst); I compute sentiment analysis with a count of negative polarity items (the more the worst).
Another important index we implemented is the Deep Conceptual index, which is obtained by considering the proportion of Abstract vs Concrete words contained in the poem. This index is then multiplied with the Propositional Semantic Density which is obtained at sentence level by computing how many non verbal, and amongst the verbal, how many non inflected verbal chunks there are in a sentence.

3 Rhetoric Devices, Metrical and Prosodic Structure

The second module takes care of rhetorical devices, metrical structure and prosodic structure. This time the file is read on a verse by verse level by simply collecting strings in a sequence and splitting verses at each newline character. In a subsequent loop, whenever two newlines characters are met, a stanza is computed. In order to compute rhetorical and prosodic structure we need to transform each word into its phonetic counterpart, by accessing the transcriptions available in the CMU dictionary. The Carnegie Mellon Pronouncing Dictionary is freely available online and includes American English pronunciation[3]. Kaplan reports the existence of another dictionary which is however no longer available.[4] The version of the CMU dictionary they are referring to is 0.4 and is the version based on phone/phoneme transcription.
Kaplan & Blei in their longer paper specifies that “No extra processing is done to determine pronunciation ... so some ambiguities are resolved incorrectly.” [9:42]. In fact what they are using is the phoneme version of the dictionary and not the syllabified one, which has also been increased by new words. We had available a syllable parser which was used to build the VESD database of English syllables [1]. So we started out with a much bigger pronunciation dictionary which covers 170,000 entries approximately.
Remaining problems to be solved are related to ambiguous homographs like “import” (verb) and “import” (noun) and are treated on the basis of their lexical category derived

[3] It is available online at <http://www.speech.cs.cmu.edu/cgi-bin/cmudict/>.
[4] Previously, data for POS were merged in from a different dictionary (MRC Psycholinguistic Database, <http://lcb.unc.edu/software/multimrc/multimrc.zip>, which uses British English pronunciation)

from previous tagging and Out Of Vocabulary Words (OOVW). As happens in Kaplan's system, if a word is not found in the dictionary, we also try different capitalizations, as well as breaking apart hyphenated words, and then we check at first for 'd, 's, and s' endings and try combining those sounds with the root word. The simplest case is constituted by differences in spelling determined by British vs. American pronunciation. This is taken care of by a dictionary of graphemic correspondances. However, whenever the word is not found we proceed by morphological decomposition, splitting at first the word from its prefix and if that still does not work, its derivational suffix. As a last resource, we use an orthographically based version of the same dictionary to try and match the longest possible string in coincidence with our OOVW. Then we deal with the remaining portion of word again by guessing its morphological nature, and if that fails we simply use our grapheme-to-phoneme parser.

3.1 Computing Metrical Structure and Rhyming Scheme

After reading out the whole poem on a verse by verse basis and having produces all phonemic transcription, we look for rhetoric devices. Here assonances, consonances, allitterations and rhymes are analysed and then evaluated. We introduce an important prosodic element: we produce a prosodic model of the poem and compute duration at verse level. This is done by associating durations at syllable level. In turn, these data are found by associating phonemes into syllables with our parser, which works on the basis of the phonological criterion of syllable wellformedness. Syllable structure requires a nucleus to be in place, then a rhyme with an onset and offset[1]. Durations have been recorded by means of a statistical study, with three different word positions: beginning, middle and end position. They have also been collected according to a prosodic criterion: stressed and unstressed syllables. Each syllable has been recorded with three durational values in msec.: minimum, mean and maximum duration length, with a standard deviation. To produce our prosodic model we take mean durational values. We also select, whenever possible, positional and stress values. Of course, if a syllable duration value is not available for those parameters we choose the default value, that is unstressed. Then we compute metrical structure, that is the alternation of beats: this is computed by considering all function or grammatical words which are monosyllabic as unstressed. We associate a "0" to all unstressed syllables, and a value of "1" to all stressed syllables, thus including both primary and secondary stressed syllables.
Durations are then collected at stanza level and a statistics is produced. Metrical structure is used to evaluate statistical measures for its distribution in the poem. As can be easily gathered from our transcription, it is difficult to find verses with identical number of syllables, identical number of metrical feet and identical metrical verse structure. If we consider the sequence "01" as representing the typical iambic foot, and the iambic pentameter as the typical verse metre of English poetry, in our transcription it is easy to see that there is no line strictly respecting it. On the contrary we find trochees, "10", dactyls, "100", anapests, "001"and spondees, "11". At the end of the computation, the system is able to measure two important indices: "mean verse length" and "mean verse length in no. of feet" that is mean metrical structure.
Additional measure that we are now able to produce are related to rhyming devices. Since we intended to take into account structural internal rhyming scheme and their persistence in the poem we enriched our algorithm with additional data. These measures are then

accompanied by information derived from two additional component: word repetition and rhyme repetition at stanza level. Sometimes also refrain may apply, that is the repetition of an entire line of verse. Rhyming schemes together with metrical length, are the strongest parameters to consider when assessing similarity between two poems.
Eventually also stanza repetition at poem level may apply: in other words, we need to reconstruct the internal structure of metrical devices used by the poet. We then use this information as a multiplier. The final score is then tripled in case of structural persistence of more than one rhyming scheme; for only one repeated rhyme scheme, it is doubled. With no rhyming scheme there will be no increase in the linear count of rethorical and rhyming devices. Creating the rhyming scheme is not an easy task. We do that by a sequence of incremental steps that assign labels to each couple of rhyming line and then matches their output. To create rhyme schemes we need all last phonetic words coming from our previous analysis. We then match recursively each final phonetic word with the following ones, starting from the closest to the one that is 6 lines far apart. Each time we register the rhyming words and their distance, accompanied by an index associated to verse number. Stanza boundaries are not registered in this pass.
The following pass must reconstruct the actual final verse numbers and then produce an indexed list of couples, Verse Number-Rhyming Verse for all the verses, stanza boundaries included. Eventually, we associate alphabetic labels to the each rhyming verse starting from A to Z. A simple alphabetic incremental mechanism updates the rhyme label. This may go beyond the limits of the alphabet itself and in that case, double letter are used.
I distinguish between poems divided up into stanzas and those that have no such a structure. Then I get stanzas and their internal structure in term of rhyming labels. Eventually what I want to know is the persistence of a given rhyme scheme, how many stanza contain the same rhyme scheme and the length of the scheme. A poem with no rhyme scheme is much poorer than a poem that has at least one, so this needs to be evaluated positively and this is what I do. In the final evaluation, it is possible to match different poems on the basis of their rhetorical and rhyming devices, besides their semantic and conceptual indices.
Parameters related to the Rhyming Scheme (RS) contribute a multiplier to the already measured metrical structure which as we already noted is extracted from the following counts: a count of metrical feet and its distribution in the poem; a count of rhyming devices and their distribution in the poem; a count of prosodic evaluation based on durational values and their distribution. Now the RS is yet another plane or dimension on the basis of which a poem is evaluated. It is based on the regularity in the repetition of a rhyming scheme across the stanzas or simply the sequence of verses in case the poem is not divided up into stanzas. We don't assess different RSs even though we could: the only additional value is given by the presence of a Chain Rhyme scheme, that is a rhyme present in one stanza which is inherited by the following stanza. Values to be computed are related to the Repetition Rate (RR), that is how many rhymes are repeated in the scheme or in the stanza: this is a ratio between number of verses and their rhyming types. For instance, a scheme like AABBCC, has a higher repetition rate (corresponding to 2) than say AABCDD (1.5), or ABCCDD (1.5). So the RR is one parameter and is linked to the length of the scheme, but also to the number of repeated schemes in the poem: RS may change during the poem and there may be more than one scheme.
Different evaluation are given to full rhymes, which add up the number of identical phones, with respect to half-rhymes which on the contrary count only half that number. The final value is obtained by dividing up the RR by the total number of lines and multiplying by

100, and then summing the same number of total lines to the result. This is done to balance the difference between longer vs. shorter poems, where longer poems are rewarded for the intrinsic difficulty of maintaining identical rhyming schemes with different stanzas and different vocabulary.

4 Conclusion and Future Work

We still have a final part of the algorithm to implement which is more complicated to do and is concerned with Modeling Poetry Reading by a TTS (Text To Speech) system. It is the intermingling of syntactic structure and rhetoric and prosodic structure into phonological structure. What remains to be done is to use syntactic information in order to "demote" stressed syllables of words included in a "Phonological Group" and preceding the Head of the group. This part of the work will have to match tokens with possible multiwords and modify consequently word level stress markers from primary "1" to secondary "2". A first prototype has been presented in[4], but more work is needed to tune prosodic parameters for expressivity rendering both at intonational and rhythmic level. The most complex element to control seems to be variations at discourse structure which are responsible for continuation intonational patterns vs. beginning of a new contour. Also emphasis is difficult to implement due to lack of appropriate semantic information.

References

1. Bacalu C., Delmonte R. (1999). Prosodic Modeling for Syllable Structures from the VESD - Venice English Syllable Database, in Atti 9° Convegno GFS-AIA, Venezia.
2. Delmonte R. (1980). Computer Assisted Literary Textual Analysis with Keymorphs and Keyroots, REVUE-Informatique et Statistique dans les Sciences humaines,1, 21-53.
3. Delmonte R. (1983). A Quantitative Analysis of Linguistic Deviation: Francis Webb, a Schizophrenic Poet, in REVUE – Informatique et Statistique dans les Sciences humaines, 19:1-4, 55-112.
4. Delmonte R. (2013). SPARSAR: a System for Poetry Automatic Rhythm and Style AnalyzeR, SLATE 2013, Demonstration Track.
5. Delmonte R. (2013), Transposing Meaning into Immanence: The Poetry of Francis Webb, in Rivista di Studi Italiani, Vol. XXX1, n° 1, 835-892.
6. Delmonte R., Sara Tonelli, Marco Aldo Piccolino Boniforti, Antonella Bristot, Emanuele Pianta (2005). VENSES – a Linguistically-Based System for Semantic Evaluation, in J. Quiñonero-Candela et al.(eds.), 2005. Machine Learning Challenges. LNCS, Springer, Berlin, 344-371.
7. Kao Justine and Dan Jurafsky. 2012. A Computational Analysis of Style, Affect, and Imagery in Contemporary Poetry. NAACL Workshop on Computational Linguistics for Literature.
8. Kaplan, D. (2006). Computational analysis and visualized comparison of style in American poetry. Unpublished undergraduate thesis.
9. Kaplan, D., & Blei, D. (2007). A computational approach to style in American poetry. In *IEEE Conference on Data Mining.*
10. Miles, J. (1967). Style and Proportion: The Language of Prose and Poetry. Little, Brown and Co., Boston.
11. Webb Francis, 1969. Collected Poems, Angus and Robertson, Sydney & London.